



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,053	10/09/2003	Chandan Mathur	1934-12-3	3240

7590 10/19/2006

Bryan A. Santarelli  
GRAYBEAL JACKSON HALEY LLP  
Suite 350  
155-108th Avenue NE  
Bellevue, WA 98004-5901

EXAMINER

HUISMAN, DAVID J

ART UNIT	PAPER NUMBER
----------	--------------

2183

DATE MAILED: 10/19/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

# Office Action Summary

Application No.

10/684,053

Applicant(s)

MATHUR ET AL.

Examiner

David J. Huisman

Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 03 August 2006.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 1-61 is/are pending in the application.
- 4a) Of the above claim(s) 25-36 and 55-61 is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-24 and 37-54 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 14 May 2004 is/are: a) ☐ accepted or b) ☒ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

## Attachment(s)

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)  | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)   | 5) <input type="checkbox"/> Notice of Informal Patent Application                       |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)<br>Paper No(s)/Mail Date <u>4/11/05 &amp; 9/19/05</u> . | 6) <input type="checkbox"/> Other: _____  |

### **DETAILED ACTION**

1. Claims 1-24 and 37-54 have been examined.

#### ***Papers Submitted***

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: IDS as received on 4/11/2005, IDS as received on 9/19/2005, and Response to Restriction Requirement as received on 8/3/2006.

#### ***Information Disclosure Statement***

3. The information disclosure statement filed on April 11, 2005, fails to fully comply with 37 CFR 1.98(a)(3) because it does not include a concise explanation of the relevance, as it is presently understood by the individual designated in 37 CFR 1.56(c) most knowledgeable about the content of the information, of the non-patent literature document entitled "Un Seul FPGA Dope Le Traitement D'Images" by Lecurieux-Lafayette, which is not in the English language. Consequently, this document has not been considered.

#### ***Election/Restrictions***

4. Claim 34 has been associated with Species I by applicant. The examiner feels that this association is improper because claim 34 focuses primarily on exception details (Species II, Fig.6) while briefly mentioning that a processor executes an application. However, all processors execute an application, and consequently, claim 34 is not directed to Species I.

Art Unit: 2183

5. Claims 35-36 and 61 have been associated with Species I and Species II by applicant.

The examiner feels that this association is improper because claims 35-36 and 61 recite only configuration details, which are associated with Species III (Fig.7). While the phrase “configuration manager” may not appear in the claims, the claims still involve configuration, which is Fig.7 (not Fig.5 or Fig.6).

6. In light of the reasoning above, the final claim groupings are as follows:

- Species I (Fig.5) - claims 1-24 and 37-54.
- Species II (Fig.6) - claims 1-3, 5-12, 14-24, 31-34, 37, 39-45, 47-54, and 58-60.
- Species III (Fig.7) - claims 1-3, 5-12, 14-30, 33-37, 39-45, 47-57, and 59-61.

7. Applicant’s election of Species I (Fig.5) in the reply filed on August 3, 2006, is acknowledged. Because applicant did not distinctly and specifically point out the supposed errors in the restriction requirement, the election has been treated as an election without traverse (MPEP § 818.03(a)).

### *Specification*

8. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

9. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

10. The disclosure is objected to because of the following informalities:

Art Unit: 2183

- On page 1, paragraph 2, and throughout the rest of the specification, please remove the attorney docket numbers of all related applications and insert corresponding application numbers or patent numbers.
- On page 15, insert a period at the end of line 2.

Appropriate correction is required.

### ***Drawings***

11. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they do not include the following reference sign(s) mentioned in the description: On page 12, paragraph 50, reference numbers 56 and 60 are mentioned but do not appear in the drawings anywhere. Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

### ***Claim Objections***

12. Claim 6 is objected to because of the following informalities: In line 3, replace "an" with --a--. Appropriate correction is required.

Art Unit: 2183

13. Claim 16 is objected to because of the following informalities: In line 3, replace “an” with --a--. Appropriate correction is required.
14. Claim 19 is objected to because of the following informalities: In lines 5-6, replace “an communication” with --a communication--. Appropriate correction is required.
15. Claim 22 is objected to because of the following informalities: In lines 7-8, replace “an communication” with --a communication--. Appropriate correction is required.
16. Claim 49 is objected to because of the following informalities: In line 2, replace “an communication” with --a communication--. Appropriate correction is required.
17. Claim 51 is objected to because of the following informalities: In line 7, replace “an” with --a--. Appropriate correction is required.

***Claim Rejections - 35 USC § 112***

18. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.
19. Claims 5, 14, 21, 24, 39, and 53-54 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.
20. Claim 5 recites the limitation "the second software object" in line 7, and the limitation “the storage location” in line 10. There is insufficient antecedent basis for these limitations in the claim. For purposes of examination, “the second software object” will be interpreted as “the second data-transfer object” and “the storage location” will be interpreted as “the buffer”.

Art Unit: 2183

21. Claim 14 recites the limitation "the published data" in lines 5, 7, and 10. There is insufficient antecedent basis for this limitation in the claim. For purposes of examination, "the published data" will be interpreted as "the retrieved data".

22. Claim 21 recites the limitation "the host processor" in line 2. There is insufficient antecedent basis for this limitation in the claim. For purposes of examination, "the host processor" will be interpreted as "the processor".

23. Claim 24 recites the limitation "the host processor" in line 2. There is insufficient antecedent basis for this limitation in the claim. For purposes of examination, "the host processor" will be interpreted as "the processor".

24. Claim 39 recites the limitation "the storage location" in line 7. There is insufficient antecedent basis for this limitation in the claim. For purposes of examination, "the storage location" will be interpreted as "the buffer".

25. Claim 53 recites the limitation "the communication buffer" in line 3. There is insufficient antecedent basis for this limitation in the claim. For purposes of examination, "the communication buffer" will be interpreted as "a communication buffer".

26. Claim 54 is rejected under 35 U.S.C. 112, 2<sup>nd</sup> paragraph, for being indefinite, because it is dependent on claim 53, and claim 53 is rejected for being indefinite.

### ***Claim Rejections - 35 USC § 102***

27. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

Art Unit: 2183

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

28. Claims 1-3, 5-6, 9-12, 14-17, 37, 39-40, 43, 45, and 47-49 are rejected under 35

U.S.C. 102(b) as being anticipated by Tamura, U.S. Patent No. 6,108,693.

29. Referring to claim 1, Tamura has taught a computing machine, comprising:

a) a first buffer. See Fig.4, component 442a, and note the first communication buffer.

b) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and operable to:

b1) execute an application, a first data-transfer object, and a second data-transfer object.

A processor executes an application, where an application is any program that causes operations to be performed by the processor. Also, first and second data transfer objects are executed. See Fig.4, components 400 and 430, respectively.

b2) publish data under the control of the application. See column 8, lines 63-64, and note that data of array A is published (i.e., generated).

b3) load the published data into the buffer under the control of the first data-transfer object. See Fig.6, and note the first data transfer object (transmit object) loads the published data into the buffer.

b4) retrieve the published data from the buffer under the control of the second data-transfer object. See Fig.6, and note that the second data transfer object (receive object) retrieves the data from the buffer.

30. Referring to claim 2, Tamura has taught a computing machine as described in claim 1.

Tamura has further taught that the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object



Art Unit: 2183

writes data to the buffer and the second object reads data from the buffer. Both objects clearly access the buffer, and consequently, both comprise code to access the buffer. The objects' code is the same in at least this respect (it causes buffer access).

31. Referring to claim 3, Tamura has taught a computing machine as described in claim 1.

Tamura has further taught the processor comprises:

a) a processing unit operable to execute the application and publish the data under the control of the application. Recall from the rejection of claim 1 that the processor executes an application and publishes data under control of the application. This execution and publishing is performed by a processing unit.

b) a data-transfer handler operable to execute the first and second data-transfer objects, to load the published data into the buffer under the control of the first data-transfer object, and to retrieve the published data under the control of the second data-transfer object. Again, recall from the rejection of claim 1 that first and second objects, for loading and retrieving from a buffer, respectively, are executed. This unit which performs this execution, loading, and retrieving is a data transfer handler.

32. Referring to claim 5, Tamura has taught a computing machine as described in claim 1.

Tamura has further taught that the processor is further operable to:

a) execute a queue object and a reader object. See column 2, lines 56-67, and note that the queue object writes to the buffer and sets the buffer from read-disabled to read-enabled. The reader object sets the buffer from write-disabled to write-enabled.

b) store a queue value under the control of the queue object, the queue value reflecting the loading of the published data into the buffer. Again, see column 2, lines 56-67, and note that by

Art Unit: 2183

storing a value indicative of the buffer being read-enabled in flag 442b (Fig.4), the system is signaling that published data has been loaded into the buffer and it can now be read.

c) read the queue value under the control of the reader object. See column 8, lines 27-29. The system must first detect whether the buffer is read-enabled, and therefore it must read the queue value. If it is read-enabled, then the buffer can be read.

d) notify the second software object that the published data occupies the buffer under the control of the reader object and in response to the queue value. See column 8, lines 29-31. If the read enable flag is set, then the published data may be read by the second data transfer object.

e) retrieve the published data from the storage location under the control of the second data-transfer object and in response to the notification. See Fig.1, component 23, and Fig.4, component 430, and Fig.6.

33. Referring to claim 6, Tamura has taught a computing machine as described in claim 1. Tamura has further taught:

a) a bus. See Fig.4 and note the bus between local memory and a single processor of the multiprocessor.

b) wherein the processor is operable to execute a communication object and to drive the retrieved data onto the bus under the control of the communication object. The abstract discloses that data is retrieved from the buffer and sent to the local memory via a bus. The object which causes this to happen is the communication object.

34. Referring to claim 9, Tamura has taught a computing machine as described in claim 1. Tamura has further taught that:

Art Unit: 2183

a) the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object writes data to the buffer and the second object reads data from the buffer. Both clearly access the buffer, and consequently, both comprise code to access the buffer. The code of both objects is the same in at least this respect (it causes buffer access).

b) the processor is operable to execute an object factory and to generate the object code under the control of the object factory. All processors execute programs. The program (object factory) will dictate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted, the first data transfer object will be invoked so that data may be transmitted.

35. Referring to claim 10, Tamura has taught a computing machine, comprising:

a) a first buffer. See Fig.4, component 442a, and note the first communication buffer.

b) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and operable to:

b1) execute first and second data-transfer objects and an application. A processor executes an application, where an application is any program that causes operations to be performed by the processor. Also, first and second data transfer objects are executed.

See Fig.4, components 400 and 430, respectively.

b2) retrieve data and load the retrieved data into the buffer under the control of the first data-transfer object. See the abstract and Fig.6, and note the first data transfer object (transmit object) retrieves data from local memory and loads the data into the buffer.

b3) unload the data from the buffer under the control of the second data-transfer object.

See Fig.6, and note that the second data transfer object (receive object) retrieves the data from the buffer.

b4) process the unloaded data under the control of the application. After the data is retrieved from the buffer, it is processed by storing it in local memory. See the abstract.

36. Referring to claim 11, Tamura has taught a computing machine as described in claim 10. Tamura has further taught that the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object writes data to the buffer and the second object reads data from the buffer. Both clearly access the buffer, and consequently, both comprise code to access the buffer (the code is the same in at least this respect).

37. Referring to claim 12, Tamura has taught a computing machine as described in claim 10. Tamura has further taught that the processor comprises:

a) a processing unit operable to execute the application and process the unloaded data under the control of the application. Recall from the rejection of claim 10 that the processor executes an application and processes unloaded data under control of the application. This execution and processing is performed by a processing unit.

b) a data-transfer handler operable to execute the first and second data-transfer objects, to retrieve the data from the bus and load the data into the buffer under the control of the first data-transfer object, and to unload the data from the buffer under the control of the second data-transfer object. Again, recall from the rejection of claim 10 that first and second objects, for

Art Unit: 2183

loading and unloading from a buffer, respectively, are executed. This unit which performs this execution, loading, and unloading is a data transfer handler.

38. Referring to claim 14, Tamura has taught a computing machine as described in claim 10.

Tamura has further taught that the processor is further operable to:

a) execute a queue object and a reader object. See column 2, lines 56-67, and note that the queue object writes to the buffer and sets the buffer from read-disabled to read-enabled. The reader object sets the buffer from write-disabled to write-enabled.

b) store a queue value under the control of the queue object, the queue value reflecting the loading of the published data into the first buffer. Again, see column 2, lines 56-67, and note that by storing a value indicative of the buffer being read-enabled in flag 442b (Fig.4), the system is signaling that published data has been loaded into the buffer and it can now be read).

c) read the queue value under the control of the reader object. See column 8, lines 27-29. The system must first detect whether the buffer is read-enabled, and therefore it must read the queue value. If it is read-enabled, then the buffer can be read.

d) notify the second data-transfer object that the published data occupies the buffer under the control of the reader object and in response to the queue value. See column 8, lines 29-31. If the read enable flag is set, then the published data may be read.

e) unload the published data from the buffer under the control of the second data-transfer object and in response to the notification. See Fig.1, component 23, and Fig.4, component 430, and Fig.6.

39. Referring to claim 15, Tamura has taught a computing machine as described in claim 10.

Tamura has further taught:

Art Unit: 2183

a) a second buffer. See Fig.4, component 442a, and Fig.6, and note that a second buffer exists.

b) wherein the processor is operable to retrieve the data from the second buffer under the control of the first data-transfer object. See Fig.6 and note that data is received from the second buffer.

And, the receiving is under the control of the first data transfer object because the first object writes the data to the buffer and sets the read-enabled status flag for that buffer so that the receiving portion may read it. See column 2, lines 56-67.

40. Referring to claim 16, Tamura has taught a computing machine as described in claim 10. Tamura has further taught:

a) a bus. See Fig.4 and note the bus between local memory and a single processor of the multiprocessor.

b) wherein the processor is operable to execute a communication object, to receive the data from the bus under the control of the communication object, and to retrieve the data from the communication object under the control of the first data-transfer object. The object which causes sending and retrieving of data on a bus (via a buffer) is the communication object. The communication object will first receive data from local memory via a bus (see the abstract and Fig.4, and note the bus coupling local memory to component 410), and after the data is written to the buffer it is retrieved. See Fig.6. The retrieval is under control of the communication object as the communication object will set the read-enable flag which allows retrieval to occur. See column 2, lines 56-67.

41. Referring to claim 17, Tamura has taught a computing machine as described in claim 10. Tamura has further taught that:

Art Unit: 2183

a) the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object writes data to the buffer and the second object reads data from the buffer. Both, however, clearly access the buffer, and consequently, both comprise code to access the buffer. The code is the same in at least this respect (it causes buffer access).

b) the processor is operable to execute an object factory and to generate the object code under the control of the object factory. All processors execute programs. The program will dictate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted, the first data transfer object will be invoked so that data may be transmitted.

42. Referring to claim 37, Tamura has taught a method comprising:

a) publishing data with an application. See column 8, lines 63-64, and note that data of array A is published (i.e., generated).

b) loading the published data into a first buffer with a first data-transfer object. See Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) loads the published data into the buffer.

c) retrieving the published data from the buffer with a second data-transfer object. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) retrieves the data from the buffer.

43. Referring to claim 39, Tamura has taught a method as described in claim 37. Tamura has further taught:

a) generating a queue value that corresponds to the presence of the published data in the buffer.

See column 2, lines 56-67, and note that a value is generated and stored in a flag register 442b

Art Unit: 2183

(Fig.4) to indicate that published data has been loaded into the buffer and it can now be read (read-enabled).

b) notifying the second data-transfer object that the published data occupies the buffer in response to the queue value. See column 8, lines 29-31. If the read enable flag is set, then the published data may be read. Setting the buffer to be read-enabled notifies the reader that it may be read.

c) wherein retrieving the published data comprises retrieving the published data from the storage location with the second data-transfer object in response to the notification. See Fig.1, component 23, and Fig.4, component 430, and Fig.6.

44. Referring to claim 40, Tamura has taught a method as described in claim 37. Tamura has further taught driving the retrieved data onto a bus with a communication object. The abstract and Fig.4 disclose that data is retrieved from the buffer and sent to the local memory via a bus. The object which causes this to happen is the communication object.

45. Referring to claim 43, Tamura has taught a method as described in claim 37. Tamura has further taught:

a) generating data-transfer object code with an object factory. All processors execute programs. The program (object factory) will dictate/generate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted or received, the data transfer objects will be invoked so that data may be transmitted and received.

b) generating the first data-transfer object as a first instance of the object code. From Fig.4, it can be seen that a first object 400 is generated. This object will access the buffer.



Art Unit: 2183

c) generating the second data-transfer object as a second instance of the object code. From Fig.4, it can be seen that a second object 430 is generated. This object will access the buffer. The two objects are the same object code in that they both access the buffer.

46. Referring to claim 45, Tamura has taught a method comprising:

a) retrieving data and loading the retrieved data into a first buffer with a first data-transfer object.

See the abstract and Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) retrieves data from local memory and loads the data into the buffer.

b) unloading the data from the buffer with a second data-transfer object. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) unloads the data from the buffer.

c) processing the unloaded data with an application. After the data is retrieved from the buffer, it is processed by storing it in local memory (note that processing is performing an operation on data, such as storing). See the abstract.

47. Referring to claim 47, Tamura has taught a method as described in claim 45. Tamura has further taught:

a) generating a queue value that corresponds to the presence of the data in the buffer. See column 2, lines 56-67, and note that a value is generated and stored in a flag register 442b (Fig.4) to indicate that published data has been loaded into the buffer and it can now be read (read-enabled).

b) notifying the second data-transfer object that the data occupies the buffer in response to the queue value. See column 8, lines 29-31. If the read enable flag is set, then the published data may be read. Setting the buffer to be read-enabled notifies the reader that it may be read.

Art Unit: 2183

c) wherein unloading the data comprises unloading the data from the buffer with the first data-transfer object in response to the notification. See Fig.1, component 23, and Fig.4, component 430, and Fig.6.

48. Referring to claim 48, Tamura has taught a method as describe in claim 45. Tamura has further taught that retrieving the data comprises retrieving the data from a second buffer with the first data-transfer object. See Fig.6 and column 2, lines 56-67. Note that for data to be received, the read-enable flag of the buffer must be set, and this is done by the first data transfer object (Fig.4, component 400). Therefore, the data is received with the first data transfer object.

49. Referring to claim 49, Tamura has taught a method as described in claim 45. Tamura has further taught:

a) receiving the data from a bus with an communication object. See the abstract and Fig.4 and note that the transmitting portion will load data from memory via a bus for transmission. This loading is done by the communication object.

b) wherein retrieving the data comprises retrieving the data from the communication object under with the first data-transfer object. See Fig.4, component 400, and note that the first data transfer object sends data to the buffer. The data that is sent is received from the executing communication object, which loads the data.

### *Claim Rejections - 35 USC § 103*

50. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person

Art Unit: 2183

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

51. Claims 4, 13, 38, and 46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura.

52. Referring to claim 4, Tamura has taught a computing machine as described in claim 1.

Tamura has not taught that the processor is further operable to execute a thread of the application and to publish the data under the control of the thread. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art.

Specifically, it is known to divide up a program into threads in order to increase efficiency by reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the processor executes a thread of the application and publishes data under control of the thread.

53. Referring to claim 13, Tamura has taught a computing machine as described in claim 10.

Tamura has not taught that the processor is further operable to execute a thread of the application and to process the unloaded data under the control of the thread. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art.

Specifically, it is known to divide up a program into threads in order to increase efficiency by reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it

Art Unit: 2183

would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the processor executes a thread of the application and processes the unloaded data under control of the thread.

54. Referring to claim 38, Tamura has taught a method as described in claim 37. Tamura has not taught that the data comprises publishing the data with a thread of the application. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art. Specifically, it is known to divide up a program into threads in order to increase efficiency by reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the processor executes a thread of the application and publishes data with the thread.

55. Referring to claim 46, Tamura has taught a method as described in claim 45. Tamura has not taught that processing the unloaded data comprises processing the unloaded data with a thread of the application. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art. Specifically, it is known to divide up a program into threads in order to increase efficiency by reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it would have been obvious to

Art Unit: 2183

one of ordinary skill in the art at the time of the invention to modify Tamura such that the processor processes the unloaded data with a thread.

56. Claims 7 and 41-42 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of Mishler, U.S. Patent No. 5,283,883.

57. Referring to claim 7, Tamura has taught a computing machine as described in claim 1. Tamura has not taught a second buffer and that the processor is operable to provide the retrieved data to the second buffer under the control of the second data-transfer object. However, Mishler has taught the concept of a DMA unit having a buffer that holds data that is to be written to memory. See Fig.6, component 280, for instance. A DMA unit is used to provide very fast data transfers between a processor and memory since the DMA unit can transfer the data without full CPU involvement (thereby allowing the CPU to perform other important tasks). See column 1, lines 28-39. Since the data is being transferred from the receiving portion of the multiprocessor to the local memory (see the abstract of Tamura), it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura to include a DMA unit having a buffer (second buffer), which buffers the data received from the first buffer under the control of the second data transfer object. One would be motivated to make such a combination because the DMA unit and its buffer would allow for faster transfer of data from the receiving portion to the local memory while allowing the CPU to focus on other tasks not involving data transfer to memory.

58. Referring to claim 41, Tamura has taught a method as described in claim 37. Tamura has not taught loading the retrieved data into a second buffer with the second data-transfer object.

Art Unit: 2183

However, Mishler has taught the concept of a DMA unit having a buffer that holds data that is to be written to memory. See Fig.6, component 280, for instance. A DMA unit is used to provide very fast data transfers between a processor and memory since the DMA unit can transfer the data without full CPU involvement. See column 1, lines 28-39. Since the data is being transferred from the receiving portion of the multiprocessor to the local memory (see the abstract of Tamura), it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura to include a DMA unit having a buffer (second buffer) which buffers the data received from the first buffer under the control of the second data transfer object. One would be motivated to make such a combination because the DMA unit and its buffer would allow for faster transfer of data from the receiving portion to the local memory while allowing the CPU to focus on other tasks not involving data transfer to memory.

59. Referring to claim 42, Tamura has taught a method as described in claim 37. Tamura has not taught generating a header for the retrieved data with the second data-transfer object and combining the header and the retrieved data into a message with the second data-transfer object. However, Mishler has taught the concept of a DMA unit receiving a message including data to be transferred to memory and write signal, a write size, and an address (the latter three collectively being the header). See Fig.4f, step 112, and column 11, lines 27-37. A DMA unit is used to provide very fast data transfers between a processor and memory since the DMA unit can transfer the data without full CPU involvement. See column 1, lines 28-39. Since the data is being transferred from the receiving portion of the multiprocessor to the local memory (see the abstract of Tamura), it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura to include a DMA unit which receives messages including data

Art Unit: 2183

and headers under control of the second data transfer object in order to perform data transfer for the processor. One would be motivated to make such a combination because the DMA unit allows for faster transfer of data from the receiving portion to the local memory while allowing the CPU to focus on other tasks not involving data transfer to memory.

60. Claims 8 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of The "Microsoft Computer Dictionary, 4<sup>th</sup> Edition," Microsoft Press, 1999 (herein referred to as Microsoft).

61. Referring to claim 8, Tamura has taught a computing machine as described in claim 1. While Tamura has further taught that messages are sent between processors (column 1, lines 51-52), Tamura has not explicitly taught that the processor is further operable to generate a message that includes a header and the retrieved data under the control of the second data-transfer object. However, Microsoft has taught that a header is an information structure that precedes and identifies the information that follows, such as a block of bytes in communications (i.e., body of a message). Headers are common in message passing and they typically include parity bits (or other error detection bits) to ensure that the data received is error-free, and the length of the data that follows so that it can be detected whether or not the entire message has been received. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura in view of Morikawa to attach a header, as taught by Microsoft, to the data sent in order to detect errors and length of the data. And, the second object, by retrieving the message, generates the message for additional processing on the receiving end.

Art Unit: 2183

62. Referring to claim 18, Tamura has taught a computing machine as described in claim 10. While Tamura has further taught that messages are sent between processors (column 1, lines 51-52), Tamura has not explicitly taught that the processor is further operable to recover the data from a message that includes a header and the data under the control of the first data-transfer object. However, Microsoft has taught that a header is an information structure that precedes and identifies the information that follows, such as a block of bytes in communications. Headers are common in message passing and they typically include parity bits (or other error detection bits), to ensure that the data received is error-free, and the length of the data that follows so that it can be detected whether or not the entire message has been received. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura to attach a header, as taught by Microsoft, to the data sent in order to detect errors and length of the data. And, clearly, the header must be attached before the data is sent so it would be under control of the first data transfer object.

63. Claims 19-24, 44, 50-51, and 53 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of Morikawa et al., U.S. Patent No. 5,909,565 (herein referred to as Morikawa).

64. Referring to claim 19, Tamura has taught a peer-vector machine comprising:

a) a buffer. See Fig.4, component 442a, and Fig.6.

b) a bus. See Fig.4 and Fig.6 and note that data must travel on a bus, otherwise it cannot be sent anywhere.



Art Unit: 2183

c) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and to the bus and operable to:

c1) execute an application, first and second data-transfer objects, and a communication object. A processor executes an application, where an application is any program that causes operations to be performed by the processor. Also, first and second data transfer objects are executed. See Fig.4, components 400 and 430, respectively. The communication object may be interpreted as the overall portion of the program that retrieves data from local memory (abstract, Fig.4), sends the data, receives the data, and sends the received data to another location.

c2) publish data under the control of the application. See column 8, lines 63-64, and note that data of array A is published (i.e., generated).

c3) load the published data into the buffer under the control of the first data-transfer object. See Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) loads the published data into the buffer.

c4) retrieve the published data from the buffer under the control of the second data-transfer object. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) retrieves the data from the buffer.

c5) drive the published data onto the bus under the control of the communication object.

As the data is retrieved from a buffer coupled to the system (Fig.6), it is driven onto a bus. Otherwise, data could not be received.

d) Tamura has not taught a pipeline accelerator coupled to the bus and operable to receive the published data from the bus and to process the received published data. However, Morikawa has

Art Unit: 2183

taught the concept of a pipeline coprocessor (Fig.4, component 202, and the bottom of Fig.5 and Fig.6 show that the coprocessor is pipelined) that receives data from a processor via a buffer, processes the data, and then passes it back via a buffer. As disclosed in Morikawa in column 1, lines 15-19, coprocessors execute special instructions at high speed, and therefore, a processor, which is not especially equipped to handle such instructions at high speed, will pass off instructions/data to the coprocessor so that it may perform the operation at a more rapid pace. This clearly speeds up the system, and as a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the portion of the system receiving the data is a pipeline accelerator (coprocessor) which performs operations at high speed for the processor.

65. Referring to claim 20, Tamura in view of Morikawa has taught a peer vector machine as described in claim 19.

a) Tamura has further taught that the processor is further operable to construct a message that includes the published data under the control of the second data-transfer object and to drive the message onto the bus under the control of the communication object. Any data that is sent to the buffer is a message. The construction of this message would at least include retrieving it from local memory and outputting it to the buffer. This is done under the control of the second data transfer object because the second object informs the sending portion of the multiprocessor that it may write to the buffer (i.e., it may send the message). See column 2, lines 56-67. And, if data is being passed from point A to point B, then it is being passed via a bus. The driving of the message on the bus is done by the "communication object."

Art Unit: 2183

b) Tamura in view of Morikawa has further taught that the pipeline accelerator is operable to receive the message from the bus and to recover the published data from the message. See Fig.4 of Morikawa and again recall that when the message is retrieved by the receiving side, the data is recovered as it is at least part of the message, and this data must be written to the local memory.

66. Referring to claim 21, Tamura in view of Morikawa has taught a peer vector machine as described in claim 19. Tamura in view of Morikawa has further taught:

a) a registry coupled to the host processor and operable to store object data. See Fig.4 of Tamura, and note that the objects 400 and 430 are software objects that include instructions.

These instructions must be stored somewhere. The storage holding them is the registry.

b) wherein the processor is operable to execute an object factory, and to generate the first and second data-transfer objects and the communication object from the object data under the control of the object factory. All processors execute programs. The program (object factory) will dictate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted, received, and driven onto a bus, the first and second data transfer objects (Tamura, Fig.4, components 400 and 430) will be invoked/generated so that data may be transmitted and the object factory will be generated to that the data may be driven onto a bus.

67. Referring to claim 22, Tamura has taught a peer vector machine comprising:

a) a buffer. See Fig.4, component 442a, and Fig.6 (note the first buffer).

b) a bus. See Fig.4 and note the bus coupling the local memory to the sending portion of the multiprocessor. Also, if data is sent from point A to point B, there must be a bus to carry the data. Consequently, there is a bus coupling transmitting portion 410 (Fig.4) to the

Art Unit: 2183

communication buffer so that data may be sent as shown in Fig.6. These two buses together is, collectively, “the bus” (i.e., wires that carry data).

c) a portion coupled to the bus and operable to generate data and to drive the data onto the bus.

See Fig.4, component 410. This portion generates data (produces data from array A (column 8, lines 63-66) and drives it on the bus from local memory so that it may be received by the transmitting portion 410. Tamura has not explicitly taught that the portion is a pipeline accelerator. However, Morikawa has taught the concept of a pipeline processor (Fig.4, component 201, and the top of Fig.5 and Fig.6 show that the processor is pipelined) that receives data from a local memory and passes the data to a coprocessor. As is known, pipelining allows for the overlapping and parallelization of instructions, thereby increasing throughput and execution speeds. Consequently, in order to increase efficiency and throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the data is generated and driven onto a bus with a pipeline accelerator, as taught by Morikawa. It should be noted that the pipeline accelerator may be considered to be the portion of the processor which deals with transfer of data. Normal processing would be done by the non-accelerator part.

d) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and to the bus and operable to:

d1) execute an application, first and second data-transfer objects, and an communication object. A processor executes an application, where an application is any program that causes operations to be performed by the processor. Also, first and second data transfer objects are executed. See Fig.4, components 400 and 430, respectively. The

communication object may be interpreted as the overall portion of the program that retrieves data from local memory (abstract, Fig.4), sends the data, receives the data, and sends the received data to another location.

d2) receive the data from the bus under the control of the communication object. The abstract discloses that data is retrieved from the buffer and sent to the local memory via a bus. The object which causes this to happen is the communication object.

d3) load the received data into the buffer under the control of the first data-transfer object. See Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) loads the published data into the buffer.

d4) unload the data from the buffer under the control of the second data-transfer object. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) retrieves the data from the buffer.

d5) process the unloaded data under the control of the application. After the data is retrieved from the buffer, it is processed by storing it in local memory. See the abstract. Processing is simply performing an operation on data, and storing data is an operation performed on data.

68. Referring to claim 23, Tamura in view of Morikawa has taught a peer vector as described in claim 22.

a) Tamura in view of Morikawa has further taught that the pipeline accelerator is further operable to construct a message that includes the data and to drive the message onto the bus. Any data that is sent to the buffer is a message. The construction of this message would at least include retrieving it from local memory and outputting it to the buffer. And, if data is being

Art Unit: 2183

passed from point A to point B, then it is being passed via a bus. The driving of the message on the bus is done by the “communication object.”

b) the processor is operable to receive the message from the bus under the control of the communication object, and recover the data from the message under the control of the first data-transfer object. . See Fig.4 and Fig.6 of Tamura and again recall that when the message is retrieved by the receiving side of the multiprocessor, the data is recovered as it is at least part of the message, and this data must be written to the local memory. Note that the receiving via a bus is done by the “communication object” and the message is recovered in part because of the first data transfer object which allows the message to be read from the buffer by setting the read-enabled flag. See column 2, lines 56-67.

69. Referring to claim 24, Tamura in view of Morikawa has taught a peer vector machine as described in claim 22. Tamura in view of Morikawa has further taught:

a) a registry coupled to the host processor and operable to store object data. See Fig.4 of Tamura, and note that the objects 400 and 430 are software objects that include instructions.

b) wherein the processor is operable to execute an object factory and to generate the first and second data-transfer objects and the communication object from the object data under the control of the object factory. All processors execute programs. The program (object factory) will dictate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted, received, and driven onto a bus, the first and second data transfer objects (Tamura, Fig.4, components 400 and 430) will be invoked/generated so that data may be transmitted and the object factory will be generated to that the data may be driven onto a bus.

Art Unit: 2183

70. Referring to claim 44, Tamura has taught a method as described in claim 37. Tamura has not taught receiving and processing the data from the second data-transfer object with a pipeline accelerator. However, Morikawa has taught the concept of a pipeline coprocessor (Fig.4, component 202, and the bottom of Fig.5 and Fig.6 show that the coprocessor is pipelined) that receives data from a processor via a buffer, processes the data, and then passes it back via a buffer. As disclosed in Morikawa in column 1, lines 15-19, coprocessors execute special instructions at high speed, and therefore, a processor, which is not especially equipped to handle such instructions at high speed, will pass off instructions/data to the coprocessor so that it may perform the operation at a more rapid pace. This clearly speeds up the system, and as a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the portion of the system receiving the data is a pipeline accelerator (coprocessor) which performs operations at high speed for the processor.

71. Referring to claim 50, Tamura has taught a method as described in claim 45. Tamura has not explicitly taught providing the data to the first data-transfer object with a pipeline accelerator. However, Morikawa has taught the concept of a pipeline processor (Fig.4, component 201, and the top of Fig.5 and Fig.6 show that the processor is pipelined) that receives data from a local memory and passes the data to a coprocessor. As is known, pipelining allows for the overlapping and parallelization of instructions, thereby increasing throughput and execution speeds. Consequently, in order to increase efficiency and throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the data is provided to the first object with a pipeline accelerator, as taught by Morikawa.

Art Unit: 2183

72. Referring to claim 51, Tamura has taught a method comprising:

a) publishing data with an application running on a processor. See column 8, lines 63-64, and note that data of array A is published (i.e., generated). It should be noted that processors execute applications, which cause processing to occur.

b) loading the published data into a buffer with a first data-transfer object running on the processor. See Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) loads the published data into the buffer.

c) retrieving the published data from the buffer with a second data-transfer object running on the processor. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) retrieves the data from the buffer.

d) driving the retrieved published data onto a bus with an communication object running on the processor. Clearly, the data is retrieved via a bus and then sent somewhere. The object which sends the data somewhere is the communication object.

e) Tamura has not taught receiving the published data from the bus and processing the published data with a pipeline accelerator. However, Morikawa has taught the concept of a pipeline coprocessor (Fig.4, component 202, and the bottom of Fig.5 and Fig.6 show that the coprocessor is pipelined) that receives data from a processor via a buffer/bus, processes the data, and then passes it back via a buffer. As disclosed in Morikawa in column 1, lines 15-19, coprocessors execute special instructions at high speed, and therefore, a processor, which is not especially equipped to handle such instructions at high speed, will pass off instructions/data to the coprocessor so that it may perform the operation at a more rapid pace. This clearly speeds up the system, and as a result, it would have been obvious to one of ordinary skill in the art at the time



Art Unit: 2183

of the invention to modify Tamura such that the portion of the system receiving the data is a pipeline accelerator (coprocessor) which performs operations at high speed for the processor.

73. Referring to claim 53, Tamura has taught a method comprising:

a) generating data and driving the data onto a bus. See column 8, lines 63-64, and note that data of array A is generated and if the data is to travel throughout the processor, then it must be driven on a bus.

b) receiving the data from the bus with the communication object. Data must be received on a bus before it can be loaded anywhere (i.e., into the buffer as shown in Fig.6). Therefore, the object receiving data from the bus is a communication object.

c) loading the received data into a buffer under with a first data-transfer object. See Fig.6, and note the first data transfer object (transmit object (Fig.4, at least component 400)) loads the received data From local memory view the communication object into the buffer.

d) unloading the data from the buffer with a second data-transfer object. See Fig.6, and note that the second data transfer object (receive object (Fig.4, at least component 430)) retrieves the data from the buffer.

e) processing the unloaded data with an application. After the data is retrieved from the buffer, it is processed at least by storing it in local memory. See the abstract. Processing is simply performing an operation on data, and storing is processing.

f) Tamura has not taught that the generation and driving of data is performed by a pipeline accelerator. However, Morikawa has taught the concept of a pipeline processor (Fig.4, component 201, and the top of Fig.5 and Fig.6 show that the processor is pipelined) that receives data from a local memory and passes the data to a coprocessor. As is known, pipelining allows

Art Unit: 2183

for the overlapping and parallelization of instructions, thereby increasing throughput and execution speeds. Consequently, in order to increase efficiency and throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the data is generated and driven on a bus by a pipeline accelerator, as taught by Morikawa.

74. Claims 52 and 54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of Morikawa and further in view of Microsoft.

75. Referring to claim 52, Tamura in view of Morikawa has taught a method as described in claim 51.

a) While Tamura has taught generating a message (a message is simply the data transmitted and received), Tamura in view of Morikawa has not taught generating a message that includes a header and the published data with the second data-transfer object. However, Microsoft has taught that a header is an information structure that precedes and identifies the information that follows, such as a block of bytes in communications (i.e., body of a message). Headers are common in message passing and they typically include parity bits (or other error detection bits), to ensure that the data received is error-free, and the length of the data that follows so that it can be detected whether or not the entire message has been received. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura in view of Morikawa to attach a header, as taught by Microsoft, to the data sent in order to detect errors and length of the data. And, the second object, by retrieving the message, generates the message for additional processing on the receiving end.

Art Unit: 2183

b) Tamura in view of Morikawa has further taught that driving the data onto the bus comprises driving the message onto the bus with the communication object. The data is retrieved and then sent somewhere via a bus. The object which sends the data somewhere is the communication object.

c) Tamura in view of Morikawa has further taught that receiving and processing the published data comprises receiving the message and recovering the published data from the message with the pipeline accelerator. Again, recall Fig.4 of Morikawa where a message is received and the data in the message is processed by the accelerator (i.e., coprocessor).

76. Referring to claim 54, Tamura in view of Morikawa has taught a method as described in claim 53.

a) While Tamura has taught generating a message (a message is simply the data transmitted and received), Tamura in view of Morikawa has not taught that generating the data comprises constructing a message that includes a header and the data with the pipeline accelerator. However, Microsoft has taught that a header is an information structure that precedes and identifies the information that follows, such as a block of bytes in communications (i.e., body of a message). Headers are common in message passing and they typically include parity bits (or other error detection bits), to ensure that the data received is error-free, and the length of the data that follows so that it can be detected whether or not the entire message has been received. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura in view of Morikawa to have the pipeline accelerator attach a header, as taught by Microsoft, to the data sent in order to detect errors and length of the data.

b) Tamura in view of Morikawa has further taught that driving the data comprises driving the message onto the bus with the pipeline accelerator. The data is retrieved and then sent somewhere (driven) via a bus. The portion which does the driving is part of the pipeline accelerator.

c) Tamura in view of Morikawa has further taught that receiving the data comprises receiving the message from the bus with the communication object. Again, recall Fig.6 of Tamura and Fig.4 of Morikawa. Data is received from the buffer via a bus and the portion which does receiving is the "communication object".

d) Tamura in view of Morikawa has further taught recovering the data from the message with the first data transfer object. Since the message includes the data and a header, the data would have to be extracted. The portion doing the extraction is the "first data transfer object".

### *Conclusion*

77. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Applicant is reminded that in amending in response to a rejection of claims, the patentable novelty must be clearly shown in view of the state of the art disclosed by the references cited and the objections made. Applicant must also show how the amendments avoid such references and objections. See 37 CFR § 1.111(c).

Hansen, "The Architecture of Concurrent Programs," 1977, has shown an example of objects and buffers being used to transfer data from a card reader to a line printer.

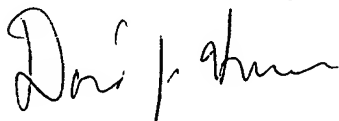
Art Unit: 2183

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

DJH  
David J. Huisman  
October 15, 2006

A handwritten signature in black ink, appearing to read "David J. Huisman", is written below the typed name.